NESNEYE TABANLI PROGRAMLAMA - II
Ders2-Yapıcılar ve Kapsülleme

Öğr. Gör. Dr. Murat ASLANYÜREK

Kırklareli Üniversitesi

Pınarhisar MYO

Yapıcılar(Constructor)

- Sınıftan bir nesne oluşturulduğunda derleyici otomatik olarak bir yapıcı metod çalıştırır. Bu metod oluşturulan her nesne için çağırılır. Yapıcılar, nesnenin ilk oluşturulduğunda alması gereken değerleri, kullanacağı metodları belirler.
  - ***Yapıcıların adı, sınıfın adı*** *ile aynı olmak zorundadır.*
  - *Yapıcı metodlar geriye herhangi bir değer döndürmezler.*
  - *Yapıcı metodlar aşırı yüklenebilir.*
  - *Her sınıf için default cunstructor(yapıcı) vardır. Bunlar herhangi bir parametre almazlar.*(Eğer biz bunu kullanmak istemiyorsak, kendimiz bir yapıcı metod yazmamız gerekir.)

Örnek: O1_Yapıcılar.java

```java
package Yapicilar_Kapsülleme;

public class O1_Yapicilar {
    int deger1, deger2;

    //Varsayılan yapıcı, sınıf adı ile aynıdır. Parametresizdir.
    public O1_Yapicilar()
    {
        deger1=5;
    }
    public  O1_Yapicilar(int sayi)
    {
        deger2=sayi;

    }
    public static void main(String[] args)
    {
        O1_Yapicilar nesne1 = new O1_Yapicilar();

        // yapiciya 10 parametresi gönderildi.
        O1_Yapicilar nesne2 = new O1_Yapicilar(10);

        System.out.println("İlk Nesnenin Degeri:"+nesne1.deger1);
        System.out.println("Son Nesnenin Degeri:"+nesne2.deger2);
    }
}
```

Console Problems Debug Shell Search

```
<terminated> O1_Yapicilar [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (1
İlk Nesnenin Degeri:5
Son Nesnenin Degeri:10
```

Yapıcıların Aşırı Yüklenmesi

- Yapıcı metotlar aşırı yüklenebilir. Bir metodun aşırı yüklenmesi demek aynı isimle o metotu birden fazla kullanmak demektir. Ancak aynı isimli metotları kullanırken parametre tiplerinin, sayılarının ve yerlerinin farklı olması gerekir.
- **Not**: *yapıcı metotların ismi sınıf ismiyle aynı olmak zorunda.*

Örnek: O2_YapicilarAsiriYükleme.java

```java
package Yapicilar_Kapsülleme;

public class O2_YapicilarAsiriYükleme {
    void metod1(){
        System.out.println("Parametre almayan metord çağrıldı");
    }
    void metod2(int sayi1)
    {
        System.out.print("Parametre alan metord çağrıldı");
    }
    public O2_YapicilarAsiriYükleme(){
        metod1();
    }
    public O2_YapicilarAsiriYükleme(int sayi1){
        metod2(sayi1);
    }
    public static void main(String[] args)
    {
        O2_YapicilarAsiriYükleme nesne1 = new O2_YapicilarAsiriYükleme();
        O2_YapicilarAsiriYükleme nesne2 = new O2_YapicilarAsiriYükleme(10);
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O2_YapicilarAsiriYükleme [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (1

```
Parametre almayan metord çağrıldı
Parametre alan metord çağrıldı
```

this Anahtar Kelimesi

- **this** anahtar kelimesi, o anda işlem yapan nesnenin referansını döndürür. Bu referans ile nesne değişkenlerine erişilir.
- **this** anahtar kelimesi sayesinde bir metod çağrıldığında, o metodun hangi nesne tarafından çağrıldığı bilinebilir.
- **this** anahtar kelimesi, isim çakışmalarında karışıklığı önlemek için de kullanılır.
- **this** anahtar kelimesi kullanılarak nesne değişkeninin kullanılacağı belirtilir.

Örnek: O3_thisA ve O3_thisB

```java
package Yapicilar_Kapsülleme;

public class O3_thisA {
    String ad;
    public void degistir(String ad){
        ad=ad;
    }
    public static void main(String[] args)
    {
        O3_thisA t = new O3_thisA();
        t.degistir("Çağan");
        System.out.print(t.ad);
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O3_thisA [Java Application] C:\Program Files\Java\j

```
null
```

```java
package Yapicilar_Kapsülleme;

public class O3_thisB {
    String ad;
    public void degistir(String ad){
        this.ad=ad;
    }
    public static void main(String[] args)
    {
        O3_thisB t = new O3_thisB();
        t.degistir("Çağan");
        System.out.print(t.ad);
    }
}
```

Console | Problems | Debug Shell | Search

<terminated> O3_thisB [Java Application] C:\Program Files\Java\jdk-13.0.1

```
Çağan
```

this Anahtar Kelimesi

- **this** anahtar kelimesi bir metottan geriye değer olarak döndürülebilir.
- **this** anahtar kelimesinin bir başka kullanım amacı da aynı sınıf içerisindeki başka bir yapıcıyı çağırmaktır.

```
1  package Yapicilar_Kapsülleme;
2
3  public class O4_this {
4      String ad;
5      String soyad;
6      public O4_this(String ad){
7          this(ad,"ASLANYÜREK");
8      }
9      public O4_this(String ad, String soyad){
10         System.out.println(ad+"  "+soyad);
11     }
12     public static void main(String[] args)
13     {
14         O4_this t = new O4_this("Çağan");
15
16     }
17 }
```

Console ⊠ Problems Debug Shell Search

<terminated> O4_this [Java Application] C:\Program Files\Java\jdk-13

Çağan ASLANYÜREK

Örnek:O5_Yapicilar_this.java

```java
package Yapicilar_Kapsülleme;

public class O5_Yapicilar_this {
    int sinif;
    String program;
    public O5_Yapicilar_this() {
        this(7);
        System.out.println("Merhaba");

    }
    public O5_Yapicilar_this(int sinif, String program) {
        this.sinif = sinif;
        this.program = program;
        System.out.println("Hoşgeldiniz");
        System.out.println("Java ile programlama");
    }
    public O5_Yapicilar_this(int sinif) {
        this(sinif, "Bilgisayar Programcılığı");
        this.sinif = sinif;
        System.out.println("Nesne Tabanlı Programlama II");
        System.out.println("PMYO ");
    }
    public void siparisGoster() {
        System.out.println("Sınıfı= " + sinif + "  Programı =  " + program);
    }

    public static void main(String[] args) {
        O5_Yapicilar_this t = new O5_Yapicilar_this();
        t.siparisGoster();
    }
}
```

Kapsülleme(Encapsulation)

- Kapsülleme, bir sınıfın içerisindeki metod ve değişkenlerin korumasıdır.
- Bünyesindeki metod ve değişkenlere dışarıdan erişmenin sakıncalı olduğu durumlarda kapsülleme kullanılır.
- Kapsülleme sayesinde sınıf üyeleri dış dünyadan gizlenir ve bu gizliliğin derecesi erişim belirleyiciler tarafından tanımlanır.

Kapsülleme(Encapsulation)

Erişim Belirleyiciler

- Sınıf elemanlarında erişimi sınırlandıran anahtar kelimelerin genel adıdır. (***public, private, protected***). Anahtar kelimeleri kullanmazsak default anahtar kelime kullanılır. Kapsüllemenin temelini bu erişim belirleyiciler oluşturur.

- **public**: Bir elemanı public olarak tanımlarsak bu elemana herkes ulaşabilir.
- **protected**: Bir sınıfın protected olarak tanımlanmış elamanına o sınıftan, o sınıftan türetilmiş sınıflardan ve o sınıf ile aynı pakatte bulunan sınıflar erişebilir.
- **private**: private olarak tanımlanmış sınıf elamanlarına, yalnızca o sınıftan erişilebilir. O sınıftan türetilen sınıflardan veya aynı pakatte bulunan sınıflardan erişilemez.
- **Default**: Herhangi bir erişim belirleyici tanımlanmamışsa, default olarak kabul edilir. Default tanımlanmış sınıf elamanlarına, o sınıftan ve aynı pakatteki sınıftan erişilebilir.

```java
package Yapicilar_Kapsülleme;

public class O6_Kapsulleme_A {
    private void kasa()
    {
        System.out.println("Gizli Kasaya Erişildi");
    }
    private String belge="Gizli Belgeye Erişildi";
    public static void main(String[] args)
    {
        O6_Kapsulleme_A anne = new O6_Kapsulleme_A();
        anne.kasa();
        System.out.print(anne.belge);
    }
}
```

Console Problems Debug Shell Search

<terminated> O6_Kapsulleme_A [Java Application] C:\Program Files\Java\jdk-1

```
Gizli Kasaya Erişildi
Gizli Belgeye Erişildi
```

```java
package Yapicilar_Kapsülleme;

public class O6_Kapsulleme_B {
    public void kasa()
    {
        System.out.println("Kasaya Erişildi");
    }
    protected String belge="Korumalı Belgeye Erişildi";
```

```java
package Yapicilar_Kapsülleme;

public class O6_Kapsulleme_C {
    public static void main(String[] args)
    {
        O6_Kapsulleme_B anne = new O6_Kapsulleme_B();
        anne.kasa();
        System.out.print(anne.belge);
    }

}
```

Console Problems Debug Shell Search

<terminated> O6_Kapsulleme_C [Java Application] C:\Program Files\Java\jdk-13.0.1

```
Kasaya Erişildi
Korumalı Belgeye Erişildi
```

```java
package Yapicilar_Kapsülleme;

public class O6_Kapsulleme_D {
     void kasa()
    {
        System.out.println("Kasaya Erişildi");
    }
    private String belge="Korumalı Belgeye Erişildi";
}
```

```java
package Yapicilar_Kapsülleme;

public class O6_Kapsulleme_C {
    public static void main(String[] args)
    {
        O6_Kapsulleme_D anne = new O6_Kapsulleme_D();
        anne.kasa();
        System.out.print(anne.belge);
    }
}
```

```java
package Sinif_Nesne_Kavrami;

public class Kapsulleme {
     void kasa()
    {
        System.out.println("Kasaya Erişildi");
    }
    protected String belge="Korumalı Belgeye Erişildi";
    private int x=5;
}
```

```java
package Yapicilar_Kapsülleme;
import Sinif_Nesne_Kavrami.Kapsulleme;

public class O7_Kapsulleme {
    public static void main(String[] args)
    {
        Kapsulleme nesne = new Kapsulleme(); // Nesne Olustu
        nesne.kasa();  // erişim Yok
        System.out.print(nesne.belge); // Erişim Yok
    }
}
```

Kapsülleme(Encapsulation)
Getter ve Setter Metodlar

- Getter ve Setter metodlar, **private** elemanlara başka sınıflardan **kontrollü bir şekilde** erişmeyi sağlar. Bu metodlar sayesinde private olan üyeler okunabilir veya değerleri değiştirilebilir.
- Okuyabilmek için **getter**, yazabilmek için **setter** metodlar kullanılır.
- Getter metodlar, okuma yaptığı için **parametre almazlar**. Setter metodlar ise private üyelere değer atadığı için **parametre alırlar.**

```java
package Yapicilar_Kapsülleme;

public class O8_Getter_Setter {
    private int deger;
    private String ad;

    public int getDeger(){  // getter metod
        return deger;
    }
    public void setDeger(int deger){    // setter metod
        this.deger=deger;
    }
    public String getAd(){  //getter metod
        return ad;
    }
    public void setAd(String ad){   // setter metod
        this.ad= ad;
    }
}
```

```java
package Yapicilar_Kapsülleme;

public class O8_Getter_Setter_main {
    public static void main(String[] args)
    {
        O8_Getter_Setter n1 = new O8_Getter_Setter();
        //n1.deger=2020;  // Erişim yok
        //n1.ad="KLU PMYO"; // Erişim yok

        n1.setAd("KLU PMYO");
        System.out.println(n1.getAd());
        n1.setDeger(2020);
        System.out.print(n1.getDeger());
    }

}
```

static Anahtar Kelimesi

- **static** anahtar kelimesi, kısaca sınıftaki alanlara nesne oluşturmadan erişmeyi sağlar.
- Java ‘da her nesne için ayrı ayrı tutulan alanlar değil de bir sınıf için ortak tutulan alanlar oluşturmak istersek, bunun için yapmamız gereken **static** anahtar kelimesi kullanmaktır.
- Oluşturacağımız alanların başına **static** anahtar kelimesi getirerek oluşturacağımız değişkenler, sınıf değişkenleri olarak adlandırılır. Bu alanlara sınıftan bir nesne oluşturmadan doğrudan sınıf adı ile erişim yapılabilir.

Static Metodlar

- Static metodlara en iyi örnek Math sınıfının metodlarıdır. Örneğin; Math sınıfının sqrt() fonksiyonunu kullanabilmek için bu sınıftan bir nesne oluşturmaya gerek yoktur. Doğrudan Math.sqrt(); fonksiyonunu kullanarak karekök aldırabiliriz. Bunun sebebi Math sınıfındaki tüm metodların static olarak tanımlanmasıdır.
- **Not:** *Static metodlar içerisinden static olmayan bir değişkene erişilemez ve static olmayan bir metod çağrılamaz. Fakat bunun tam tersi olabilir. Yani static olmayan bir metod içerisinden, static olan bir değişkene erişilebilir ve static olan bir metod çağrılabilir.*

Static İmport

- Static olan metodlara sınıf adını yazmadan erişmek istiyorsak static import yöntemi kullanılır.

- İmoprt kelimesinden sonra static getirerek Math sınıfının bütün metodlarını ekledik. Artık Math sınıfının bütün metodlarına doğrudan erişilebilir.

```
1  package Yapicilar_Kapsülleme;
2
3  public class O10_static_import {
4      public static void main(String[] args) {
5
6          System.out.print("5 üzeri 2=  "+ pow(5,2));
7      }
8
9  }
```

```
1  package Yapicilar_Kapsülleme;
2
3  import  java.lang.Math.*;
4
5  public class O10_static_import {
6      public static void main(String[] args) {
7
8          System.out.print("5 üzeri 2=  "+ pow(5,2));
9      }
10
11 }
```

```
1  package Yapicilar_Kapsülleme;
2
3  import static java.lang.Math.*;
4
5  public class O10_static_import {
6      public static void main(String[] args) {
7
8          System.out.print("5 üzeri 2=  "+ pow(5,2));
9      }
10
11 }
```

Console Problems Debug Shell Search

<terminated> O10_static_import [Java Application] C:\Program Files\Java\jdk

5 üzeri 2= 25.0

```java
package Yapicilar_Kapsülleme;

public class O9_static_Kisiler {
    private String ad;
    private String soyad;
    private String cinsiyet;
    public static int sayi=0;

    public O9_static_Kisiler(String ad, String soyad, String cinsiyet){
        this.ad=ad;
        this.soyad=soyad;
        this.cinsiyet=cinsiyet;
        sayi++;
    }
}
```

```java
package Yapicilar_Kapsülleme;

public class O9_static_main {
    public static void main(String[] args) {
        System.out.println("Başlangıçtaki kişi sayısı="+O9_static_Kisiler.sayi);
        //sayi degiskenine sınıf adi ile eristik
        O9_static_Kisiler kisi1 = new O9_static_Kisiler("Mehmet", "Kirazlı", "Erkek");
        O9_static_Kisiler kisi2 = new O9_static_Kisiler("Mehmet", "Pehlivan", "Erkek");
        O9_static_Kisiler kisi3 = new O9_static_Kisiler("Faruk", "Kalkan", "Erkek");

        System.out.print("Nesneler oluşturulduktan sonraki kişi sayısı="+kisi1.sayi);
    }

}
```

Console Problems Debug Shell Search

<terminated> O9_static_main [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (19 Eki 2020 11:16:51)

```
Başlangıçtaki kişi sayısı=0
Nesneler oluşturulduktan sonraki kişi sayısı=3
```

```java
package Yapicilar_Kapsülleme;

public class O9_static_main {
    public static void main(String[] args) {
        System.out.println("Başlangıçtaki kişi sayısı="+O9_static_Kisiler.sayi);
        //sayi degiskenine sınıf adi ile eristik
        O9_static_Kisiler kisi1 = new O9_static_Kisiler("Murat", "Kirazlı", "Erkek");
        O9_static_Kisiler kisi2 = new O9_static_Kisiler("Mehmet", "Pehlivan", "Erkek");
        O9_static_Kisiler kisi3 = new O9_static_Kisiler("Faruk", "Kalkan", "Erkek");

        System.out.print("Nesneler oluşturulduktan sonraki kişi sayısı="+O9_static_Kisiler.sayi);
    }

}
```

Console Problems Debug Shell Search

<terminated> O9_static_main [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (19 Eki 2020 11:19:58)

```
Başlangıçtaki kişi sayısı=0
Nesneler oluşturulduktan sonraki kişi sayısı=3
```

KAYNAKLAR

- KİRAZLI, M., ‘‘ Java 7:Yeni Başlayanlar için’’, Kodlab, 8.Baskı, Ekim,2015.
- https://javaplanet.wordpress.com/2017/03/27/javada-sinifclass-yapisi/
- http://teknokafe.net/2017/09/19/java-oop-deger-ve-referans-tipleri-yapicilar-constructor/

Metin eklemek için tıklayın